Our Case No.10519/110

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
APPLICATION FOR UNITED STATES LETTERS PATENT

INVENTORS: CHRISTOPHER S. MOORE
DEREK J. BOSCH
DANIEL C. STEERE
J. JAMES TRINGALI

TITLE: METHOD FOR DELETING STORED DIGITAL DATA FROM WRITE-ONCE MEMORY DEVICE

ATTORNEY: Joseph F. Hetz
BRINKS HOFER GILSON & LIONE
P.O. BOX 10395
CHICAGO, ILLINOIS 60610
(312) 321-4719

METHOD FOR DELETING STORED DIGITAL DATA FROM WRITE-ONCE MEMORY DEVICE

CROSS-REFERENCE TO RELATED APPLICATION

This application is a continuation of U.S. application serial number 09/638,439, which is hereby incorporated by reference herein.

BACKGROUND

The present invention relates to write-once memory devices, and in particular to methods for deleting stored digital data from such devices.

Non-volatile memory is becoming standard in many products such as digital cameras and digital audio players, and write-once memory devices offer the advantage of low manufacturing costs. Upcoming copy protection standards such as the Secure Digital Media Interface require memory contents to be erased. However, erasing digital files is not possible with write-once memories, since a write-once memory by definition cannot be restored to its original, unwritten state once a file has been written into the memory.

BRIEF SUMMARY

The preferred embodiments described below operate with a write-once memory that includes a plurality of memory cells. The memory stores a digital file, and for this reason some of the memory cells are in an original, unprogrammed digital state, while others of the memory cells are in a programmed digital state. In response to a received delete command associated with the stored digital file, the embodiments described below over-write at least a portion of the stored digital file with a destructive pattern. This destructive pattern switches at least some of the memory cells associated with the digital file from the original, unprogrammed state to the programmed state. For example, some or all of the memory cells associated with the stored file can be over-written with the destructive pattern (111111).

This section has been provided by way of general introduction, and it is not intended to limit the scope of the following claims.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of a data storage system and memory device that implement a preferred embodiment of this invention.

Figure 2 is a flow chart of a method implemented by the system of Figure 1.

Figures 3a, 3b and 3c are schematic diagrams of a plurality of memory cells prior to storage of a digital file, subsequent to storage of a digital file, and subsequent to being over-written with a destructive pattern, respectively.

Figure 4 is a schematic diagram of a plurality of memory cells associated with a stored digital file, showing selected blocks of memory cells over-written with a destructive digital pattern.

Figure 5 is a flow chart of an alternative method implemented by the system of Figure 1.

Figure 6 is a flow chart of another method implemented by the system of Figure 1.

DETAILED DESCRIPTION OF THE PRESENTLY PREFERRED EMBODIMENTS

Turning now to the drawings, Figure 1 show a data storage system 10 coupled with a memory device comprising a memory array controller 20 and a 3-D write-once memory array 30.

The data storage system 10 can take any suitable form, and may for example be implemented as a digital camera, a digital audio player, a personal digital assistant, a game player, a cellular telephone, an electronic book, or a general purpose programmable computer. The memory array controller 20 and the 3-D write-once memory array 30 can be implemented as any suitable memory device. See for example the 3-D write-once memory devices shown in Johnson US Patent 6,034,882, Knall US Patent Application serial number 09/560,626, and Johnson U.S. Patent Application Serial No. 09/638,428, all three assigned to the assignee of the present invention and hereby incorporated by reference. Further details regarding alternative structures for the memory device are presented in U.S. Patent Applications

Serial Nos. 09/638,427 (now U.S. Patent No. 6,424,581) and 09/638,334, both filed on the same day as the present application, assigned to the assignee of the present application, and hereby incorporated by reference in their entirety.

As explained in the Knall and Johnson patent documents in detail, 3-D write-once memory arrays provide important economies in terms of reduced size of the memory array and associated reductions in manufacturing cost. In particular, a write-once memory array can be fabricated from memory cells, each having only two associated conductors: a wordline and a bitline. Such memory cells can be fabricated at a density of 3×10^7 memory cells per square millimeter of substrate.

In particular, the cost advantages of the present invention are important in consumer products such as digital cameras, digital audio players and electronic books. In these applications, the write-once memory array 30 is preferably field programmable, and the data storage system 10 field programs the memory array 30 with a desired digital medium, such as a file of one or a sequence of images, a text file such as that suitable for an electronic book, or a digital audio file.

In spite of its many advantages, a write-once memory array provides the disadvantage that it can only be written once. That is, the memory cells of the write-once memory array are fabricated in an initial, unprogrammed digital state, and selected memory cells can then be switched to an alternative, programmed digital state. In one example, the original, unprogrammed digital state can be identified as the Logic 0 state and the programmed digital state can be identified as the Logic 1 state (though the reverse is also possible). Because a memory cell cannot be erased once it is written, it is not possible to erase the files from a write-once memory array by restoring the associated memory cells to the Logic 0 state.

As shown in Figure 1, the 3-D write-once memory array 30 includes an allocation table and a plurality of files, file(1) through file(n). The allocation table includes pointers identifying the physical addresses of memory cells associated with each of the files. The files may be arranged in either

contiguous memory cell locations or a plurality of contiguous blocks of memory cells, where consecutive blocks are not necessary contiguous. For this reason, the allocation table can include one or more pointers for each stored file.

Figure 2 is a flow chart of a method implemented by the data storage system 10 and memory array controller 20 of Figure 1 in response to a delete command. In this example, the delete command relates to file(j). As used herein, the term "delete command" refers broadly to a command that has the effect of making a file difficult or impossible to read, regardless of how the delete command is implemented, e.g. by the various write operations described below.

As shown in block 42 of Figure 2, first the controller 20 finds the memory addresses in the memory array 30 that are associated with file(j). This is done by consulting the allocation table of the memory array 30. Next, at block 44 the controller 20 writes a destructive data pattern into some or all of the memory cells associated with file(j). This completes the delete function.

Figures 3a through 3c provide a specific example of memory cell states that can result from the method of Figure 2. Figure 3a shows selected memory cells of the write-once memory array 30 as initially fabricated, in which the memory cells are all in the Logic 0 state. Figure 3b shows these memory cells after they have been used to store a portion of file(j). Note that selected memory cells have been written from the Logic 0 state to the Logic 1 state. Figure 3c shows these same memory cells after the destructive data pattern of block 44 has been written. Note that in this case the destructive data pattern is (11111111), and thus all of the memory cells associated with file(j) have been written to the Logic 1 state. This completely obliterates file(j).

It is not essential in all embodiments that all of the memory cells associated with file(j) be over-written with the destructive data pattern. As shown in the example of Figure 4, file(j) can include a block of memory cells arranged from a first memory cell at the upper left corner of the drawing to a last memory cell at the lower right corner of the drawing. In Figure 4, selected portions of file(j) indicated by the reference numerals 54, 54', and 54" have

been over-written with the destructive data pattern, while portions of file(j) indicated by the reference numerals 52 and 52' have not been over-written. This approach is well-suited for effectively rendering stored music files unreadable by obliterating intermittent portions of the memory file.

Of course, many alternative data destruction patterns can be used, as long as they achieve the desired result of obliterating the selected portion or portions of the data file to be deleted. For example, the data sequence (01010101) can be used for the pattern, as can (00001111). Also, the pattern may be aperiodic.

Furthermore, it may be preferable in some applications to obliterate the portion of the allocation table associated with a file to be deleted rather than the data of the file itself. Such an alternative embodiment is illustrated in Figure 5. As shown at block 60, first the pointers P(j) identifying memory addresses associated with file(j) are found in the file system. Then a destructive data pattern is written over some or all of the pointers P(j) in block 62. Once the pointers P(j) have been obliterated, the associated file(j) is not readily retrieved.

The simplest implementation of the functions described above is to use software, due to the shorter time needed to implement, debug and upgrade a software system. However, hardware implementations are also possible. In this case, the controller 20 of Figure 1 includes logic that accepts a delete command and then searches the memory array 30 for the memory locations to be over-written. The hardware implementation can implement any of the algorithms described above as logic on an integrated circuit.

As another alternative, the over-write function described above can be performed in a manner that reduces the time required for the over-write operation. For example, the controller 20 can simultaneously over-write memory cells in multiple layers that share a given row and column address, thereby over-writing the memory cells in a plurality of layers of a 3-D memory array at one time rather than selecting them individually. Similarly, the controller can over-write a complete page, block or other set of memory cells simultaneously to increase the speed of the over-write operation.

In some applications it may be desirable to over-write only a small portion of memory cells associated with a particular file. For example, with the Secure Digital Media Interface used by the recording industry to control copying of digital music, one may wish to eliminate a section of the file that shows how many time a user has listened to or made copies of the data. This allows a user to reset the data without rewriting the entire data file.

One advantage of the embodiments described above is that the end user can simply use the data storage system in the conventional manner, commanding that files be deleted. Either the data storage system or the memory array controller automatically implements this delete command by over-writing as described above.

In many applications the data storage system 10 may be useable both with write-once memory arrays as described above as well as with re-writable memory arrays. For example, the memory device may be a modular, portable device that is readily connected to and disconnected from the data storage system. In this case, the method of Figure 6 has particular advantages. According to this method, first the data storage system identifies whether the installed memory device is a write-once memory device or not. This can be done by automatically sensing an electronic, mechanical, mechanic, or optical feature of the memory device. In a preferred embodiment, the memory device responds to a read command by providing an identification code identifying the memory device as a write-once memory device. Further details of implementation are described in US patent application serial no. 09/638,427 (now U.S. Patent No. 6,424,581), filed on the same day as the present application, which is hereby incorporated by reference.

Control is then transferred either to block 72 or block 74, depending upon whether the installed memory device is a write-once memory device or not. If so, a destructive data pattern is written in block 74 over some or all the memory cells associated with the file to be deleted. Any of the techniques described above can be used. In this connection, the term "memory cells associated" with a particular file is intended broadly to include both the memory cells of the allocation table that identify the physical addresses of the

memory cells storing the data of the associated file, as well as these data-storing cells themselves.

In the event the installed memory device is not a write-once memory device, the file to be deleted is deleted in the conventional manner in block 72. This is often done by clearing the pointers of the file allocation table (for example, by resetting the pointers to the Logic 0 state described above).

An advantage of this method is that the implementation of a file delete command is automatically optimized for the particular type of memory device that is currently installed. As explained above, write-once memory devices differ substantially from re-writable memory devices, and for this reason the same delete techniques are not optimal from both types of memory devices. This optimization is provided in this embodiment in a manner that is completely invisible to the end user, and thus all the advantages described above are obtained without complicating the commands used by the end user to delete files.

As used herein, the term "stored digital file" is intended broadly to encompass addressing pointers such as those of the allocation table described above, as well as the digital data itself.

"Programmed digital state" is intended broadly to encompass one or more states. For example, a programmed digital state can equal the Logic 1 state in a binary storage system, or the Logic 1 or the Logic 2 state in a three-state digital system.

The term "simultaneously" is intended broadly to encompass operations that overlap in time, whether or not they start and stop at precisely the same instant.

The term "block" as applied to memory cells refers to a contiguous set of memory cells, including a memory page for example.

The forgoing detailed description has described only a few of the many possible implementations of the present invention. For this reason, this detailed description is intended by way of illustration, and not by way of limitation. It is only the following claims, including all equivalents, that are intended to define the scope of this invention.